UNIVERSITY of STIRLING

CSCU9V5: Concurrent & Distributed Systems

Assignment 2 -- Distributed Systems

This assignment covers material presented during the lectures on distributed systems and builds upon the work in the practicals on *distributed systems*. The deadline for this assignment is

Mon, 29th November 2021 at 12:00

After submission **you will give a <u>mandatory demo</u>** to demonstrate and discuss your solution to the proposed problem in the light of the content of the module on distributed systems (second part of the module). A detailed schedule of demos will follow, starting from available slots in the week of 29th Nov.

Your submission should include:

1. a single pdf document, named **<your student number>-V5-report-2020.pdf** containing:
   i. a **short report** (roughly **up to four pages** – not considering figures and screenshots - plus a cover sheet with your student number) discussing the problem, and a description of your solution;
   ii. the **code listings** of your program. Please use appropriate report headings.
2. a zip file of your **source code** (for reference purposes only) named
   <your student number>-V5-code-2020.zip
   Please add your student number to each java file you will submit.

<u>Report</u>

The short report should summarise your solution, and include appropriate diagrams of the design, i.e. it should describe relevant classes and their relationships. Then, please describe in full detail two of the most relevant methods in your classes. A section will discuss the advanced task (if done). The report should also include screen shots of your running application. Finally, the report should end with a section explaining how complete your solution is, and if applicable, any special cases when your program is not functioning correctly.

It is important that your program code is properly commented, formatted and neatly structured for readability.

<u>Demo</u>

<u>In the demo you will be asked to present and discuss your solution, also in relation with the topics of the module,with the lecturer. The mark will consider presented code and the discussion.</u>

Marking

Considering the **submission and the demo+discussion**, you will receive marks for:

- the efficacy and correctness of the basic solution 40%
- advanced features 15%+15%
- the report 25%
- code comments and structure 5%

The demo+discussion will be an opportunity for you to show your competence on the topics of this module (*distributed systems* part) and will be considered when marking according to the scheme above, particularly as far as the first two points are concerned.

PTO

Assignment Problem

This assignment extends the theme of the labs on distributed systems. It builds on top of

- the lab on distributed systems and the slides on sockets. You can find useful information in that teaching material.
- the Distributed Mutual Exclusion (DME) algorithm seen in labs, based on a token ring with a class implementing a *ring node,* and a *start manager* class to inject a token to start off the system. A variant of that system will be developed here.

Basic Problem

You will develop *a distributed auction* that works using a token ring similar to the one you have seen in lectures and labs. The system will consist of *four nodes: three honest bidders* and one *auctioneer*. The auction works with nodes reading the current highest bid and deciding (randomly, for simplicity) whether they want to make a higher bid, overwriting the current bid. This is clearly a critical section case, which will be implemented by means of Distributed Mutual Exclusion solution via a *token ring*. The bid is saved in a file (you may remember the supermarket problem from the concurrency labs).

All nodes will be connected in a ring, and will communicate through sockets, as seen in the labs, each one listening on its own receiving port and sending messages to the port of the *next* node. Nodes are identified by their (*receiving) port number*. Nodes will be instantiated at launch time with two inputs: *their port number* and *the port number of the next node in the ring.* The auctioneer's port will be 7000, you will choose the port numbers of the other nodes to form a ring. Each node will run on an independent JVM (run on the same computer, i.e. *localhost,* in this assignment).

The *auctioneer,* a node of the ring, starts off the system by creating the file *bid.text*, writing the starting price in bid.txt (you can choose it), generating a *token* and sending it to the next node in the ring (similarly to the start manager in the *ring-based DME* example seen in the lab). As seen, the token does not need to carry information, and is important just for the synchronisation of the nodes.

Bidders may only make a bid when in possession of the token, i.e. when in the critical section, as follows:

1. wait to receive the token;
2. read the current bid from the file bid.txt (it will be in the first position of the file);
3. generate a random integer in between 0 and 1;
4. if the random number is 1 then bid a higher offer: add 10 to the current price and save it back in the first position of the file bid.txt. Print "Node NNNN: my bid is MMMM", where NNNN is the port number of the node and MMMM its bid;
5. if the random number is 0, then do not bid. Print "Node NNNN: no bid"
6. send on the token generated by the auctioneer to the next node in the ring.

PTO

When the auctioneer receives back the token all bidders have had the opportunity to bid, and the auction ends. The winner of the auction will be the node who made the highest bid (i.e. the last one to bid), which is the value left on the file "bid.txt". The winner port number can be identified by the printed messages. It is also possible that there is not a winner, if nobody bids.

At the end of an auction, all *sockets/servers* must be suitably closed. All *try-catch* used must capture relevant exceptions and print appropriate messages. Importantly, the types of exceptions that have to be caught should be suitably identified (i.e., use of generic *Exception e* is discouraged).

All nodes must also print suitable messages, identified by their port numbers, showing their activities: starting, stopping, entering and leaving a critical section, and managing the token. For instance:

```
Auctioneer: 7000 - of distributed auction is active ....
Auctioneer: 7000 - forwarded token to 7001
...
Node:       7002 - received token. My bid is 120
Node:       7002 - forwarded token to 7003
...
Auctioneer: 7000 - received token back
...
```

Important: please run each node on a different window, open four console windows (e.g. in eclipse) or use four terminals/CMD in four different windows. It must be evident that only one node at the time is playing its turn in the auction. Adapt, if needed, (random) timings to facilitate the understanding of program execution and to test your system.

All most relevant primitives, e.g. synchronization, file management, and communication, have been seen in labs (first part of the course for file management).

Develop *the socket-based distributed auction* described above, based on the skeleton code provided.

NOTE: be careful about the order for launching players and Auctioneer when starting the distributed auction …

NOTE: the basic solution and the following advanced features must be developed **in separate files in different directories,** named BasicSolution and AdvancedSolution, respectively. Both directory will be contained in the zip for the code that you will submit.

PTO

Advanced Features

You can enhance the basic solution by implementing the following two features:

1. *Multiple bids*. The auctioneer is now initialised with three parameters: the two port numbers as before and an integer representing how many bids the nodes can submit, i.e. how many rounds the token will go around the ring before the Auctioneer stops the lottery. At each round, each node can decide whether to bid or not to bid each time it receives the token, according to the given algorithm. The auctioneer will declare the auction closed after the given number of rounds. Note that the honest players must be ready to play for an indefinite number of rounds (i.e., once launched and in a ring, a player is always ready to play) and, in this case, we do not care about proper termination of Bidders.

2. *A node crashes!* Discuss in the report, in a proper section, what is going to happen if the auctioneer or a bidder crashes during a multi-round auction. Your explanation must consider the possible cases when the node can crash, for instance depending on whether the node has/does not have the token. Also, briefly comment on possible detection and recovery strategies that you might put in place for this system - a high-level description of the strategy with the indication of main needed changes will be ok here.

PTO

Note on Avoiding Academic Misconduct

Work which is submitted for assessment must be your own work. All students should note that the University has a formal policy on Academic Integrity and Academic Misconduct (including plagiarism) which can be found at https://bit.ly/37fYxPw.

Plagiarism: We are aware that assignment solutions by previous students can sometimes be found posted on GitHub or other public repositories. Do not be tempted to include any such code in your submission. Using code that is not your own will be treated as "poor academic practice" or "plagiarism" and will be penalized. To avoid the risk of your own work being plagiarised by others, do not share copies of your solution, and keep your work secure both during and after the assignment period.

Collusion: This is an individual assignment: working together with other students is not permitted. If students submit the same, or very similar work, this will be treated as "collusion" and all students involved will be penalized.

Contract cheating: Asking or paying someone else to do assignment work for you (contract cheating) is considered gross academic misconduct, and will result in termination of your studies with no award.

Submission on CANVAS

Please ensure you submit your assignment on CANVAS before **11:00am on Monday, 29th November 2021**, according to the submission guidelines reported above.

Late submission

If you cannot meet the assignment hand-in deadline and have good cause, **please contact the module coordinator by e-mail** before the deadline to explain your situation and ask for an extension through the Extension Request service on Canvas. Coursework will be accepted up to seven calendar days after the hand-in deadline (or expiry of any agreed extension), but the grade will be lowered by 3 marks per day or part thereof. After seven days the work will be deemed a non-submission.